

Méthodes de Développement Industriel (MDI)

Mathieu Acher

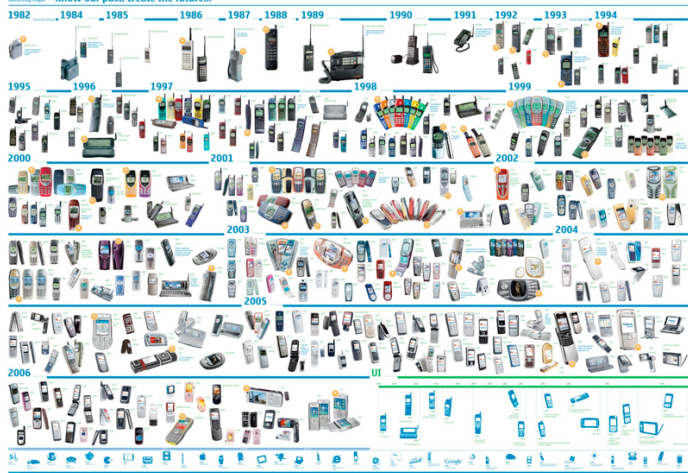
<http://www.mathieuacher.com>

Associate Professor

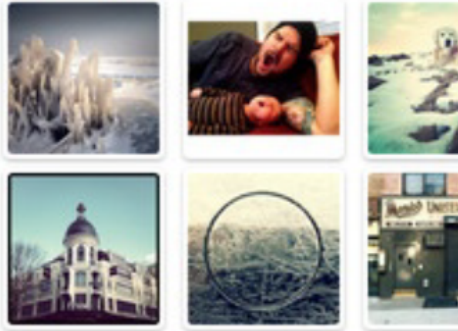
University of Rennes 1

Précédemment dans MDI

<http://mathieuacher.com/teaching/MDI/>



brayn



7 seconds ago



edroste left a comment on ernandaputra's photo
@ernandaputra wow
25 seconds ago



zachbulick and brenton_clarke like wahldesign's photo.
29 seconds ago



more than 2000 individual wires
more than 3800m cables

busses





How the customer explained it



How the Project Leader understood it



How the Analyst designed it



How the Programmer wrote it



How the Business Consultant described it



How the project was documented



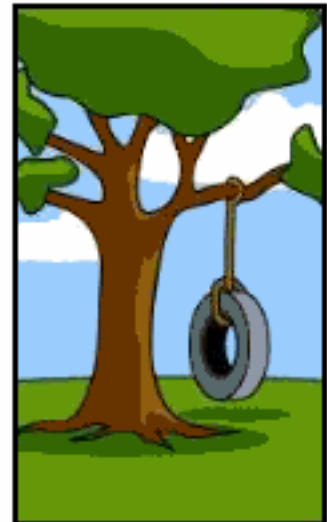
What operations installed



How the customer was billed



How it was supported



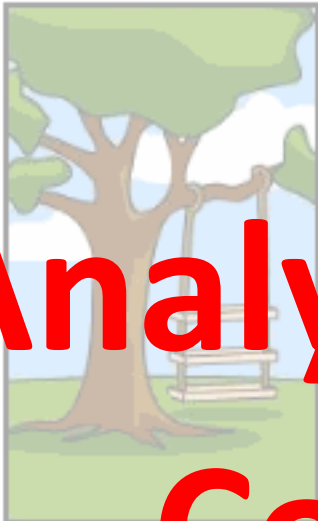
What the customer really needed

Analyse

Conception

Réalisation

Validation



How the customer explained it



How the project leader understood it



How the Analyst designed it



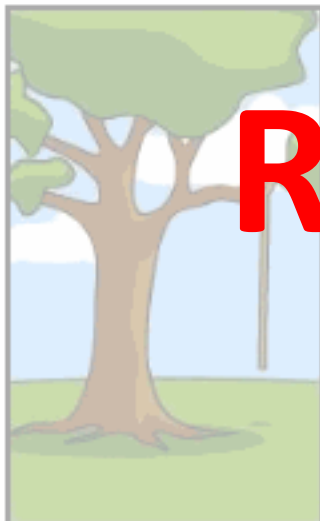
How the Programmer wrote it



How the Business Consultant described it



How the project was documented



What operations installed



How the customer was billed



How it was supported



What the customer really needed

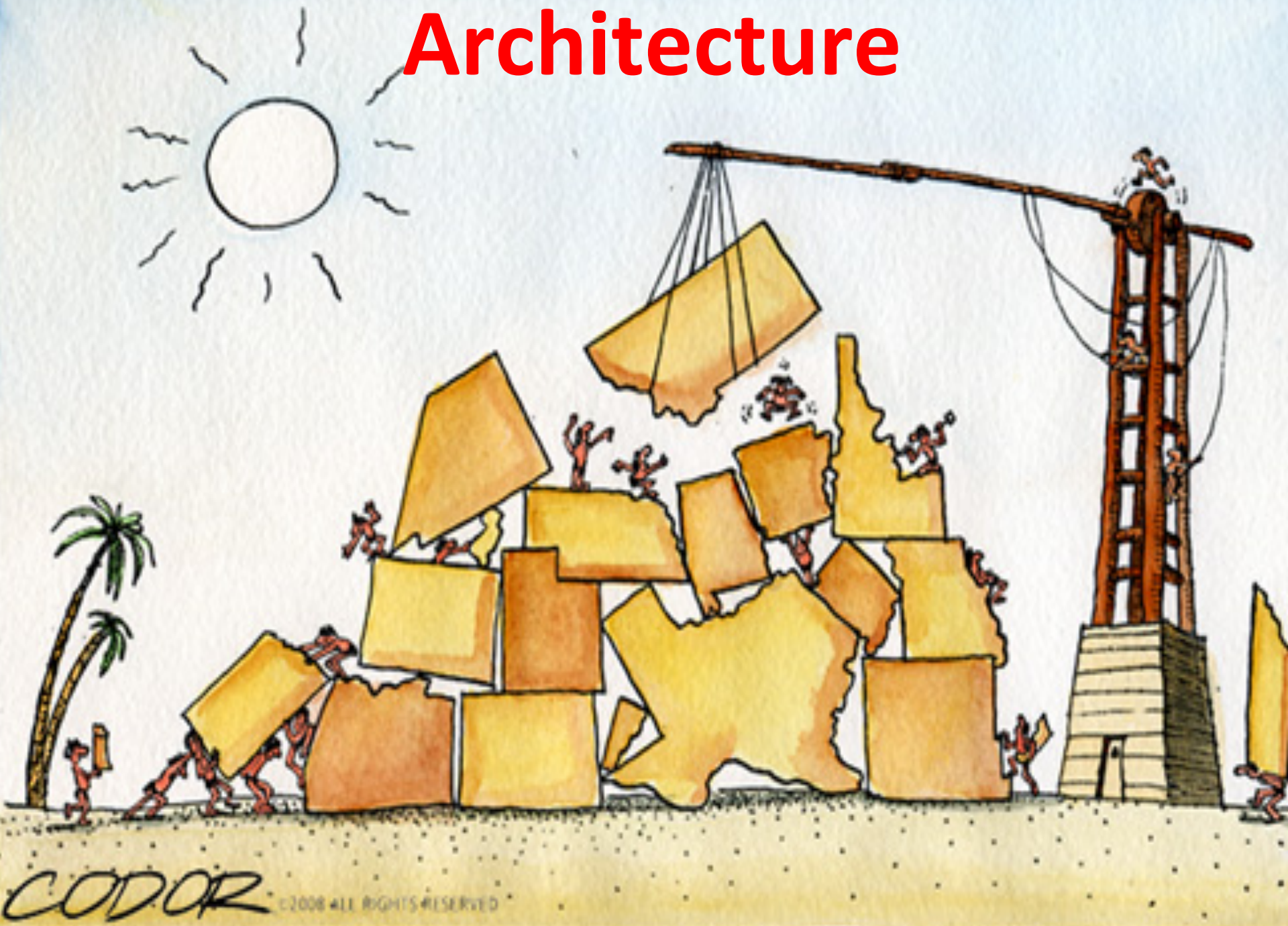


Travail d'équipe

- Organisation
 - Partage des tâches
 - Planification
 - Communication
- Code idéalement...
 - Bien conçu, modulaire, documenté
 - Maintenable, compréhensible
- Outils
 - Collaboratifs (e.g., système de versions)

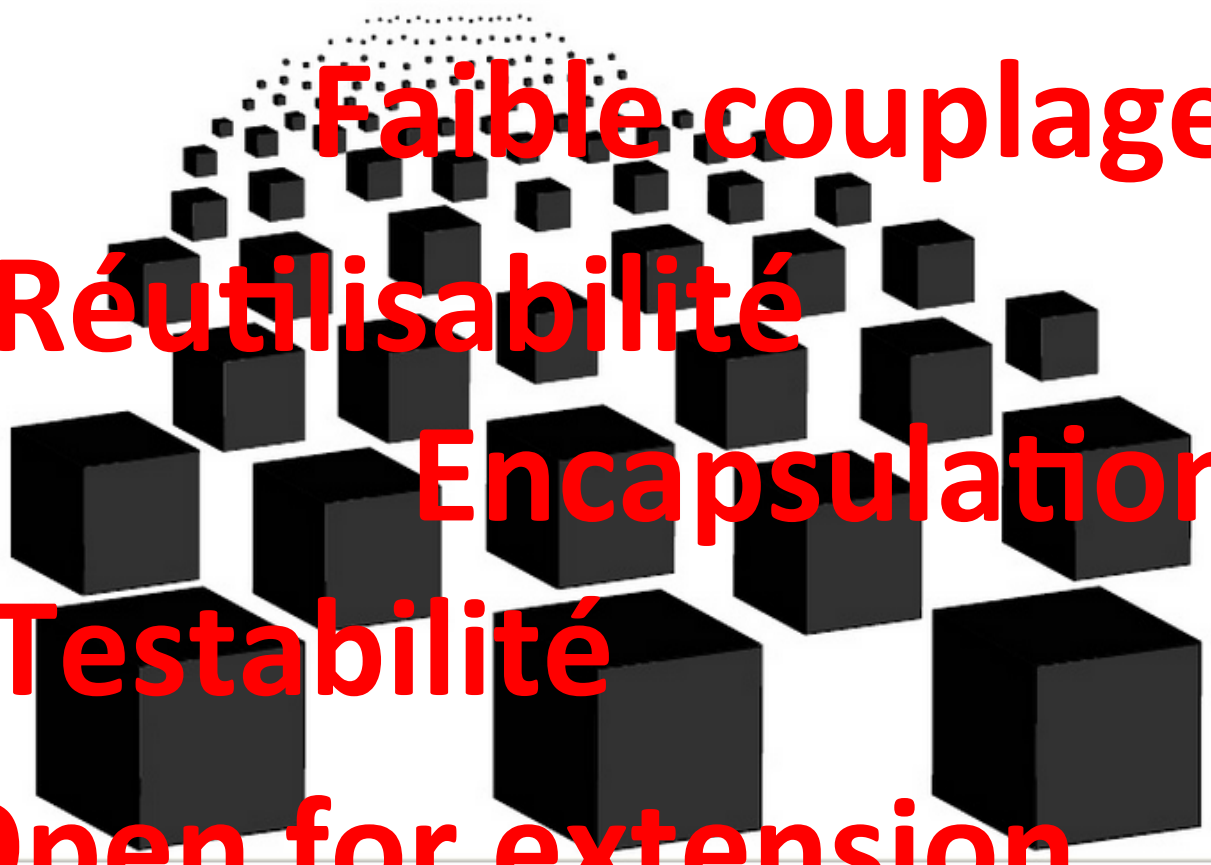


Architecture



Idéalement:
« modular black boxes »

Anti-Pattern



Faible couplage
Réutilisabilité
Encapsulation
Testabilité
Open for extension

```
ChessDay1
  main(String[]) : void
  board : PieceType[][]
  ChessDay1()
  canMove(int, int, int, int) : boolean
  displayBoard() : void
  get50MoveRulePlyCount() : int
  getCapturedPieces() : List<PieceType>
  getCapturedPieces(boolean) : List<PieceType>
  getCurrentMoveNumber() : int
  getHistory() : String
  getMaterialCount(boolean) : int
  getPossibleMoves() : List<int[][]>
  getPossibleSquares(int, int) : List<int[][]>
  getThreats(int, int) : List<int[][]>
  getUnCapturedPieces(boolean) : List<PieceType>
  initBoard() : void
  isBlackCastableKingside() : boolean
  isBlackCastableQueenside() : boolean
  isBlockable() : boolean
  isCheck() : boolean
  isCheckmate() : boolean
  isDoubleCheck() : boolean
  isEnPasantFile(int) : boolean
  isStalemate() : boolean
  isWhiteCastableKingside() : boolean
  isWhiteCastableQueenside() : boolean
  isWhiteToPlay() : boolean
  recordMove(int, int, int, int) : boolean
  redo() : boolean
  reset() : void
  setBlackCastableKingside(boolean) : void
  setBlackCastableQueenside(boolean) : void
  setWhiteCastableKingside(boolean) : void
  setWhiteCastableQueenside(boolean) : void
  undo() : boolean
```

**I don't
make
mistakes**

Testing



Des logiciels complexes...

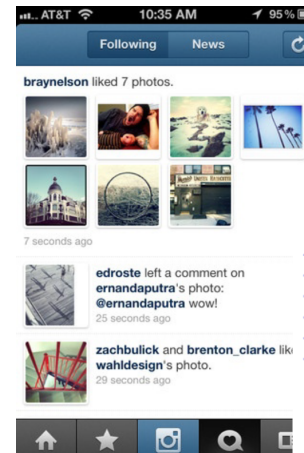
■ Logiciels de grande taille

- des millions de lignes de code
- des équipes nombreuses
- durée de vie importante
- des lignes de produits
- plateformes technologiques complexes
- évolution continue

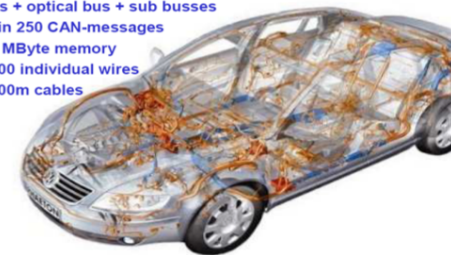
■ Logiciels complexes

■ Logiciels critiques

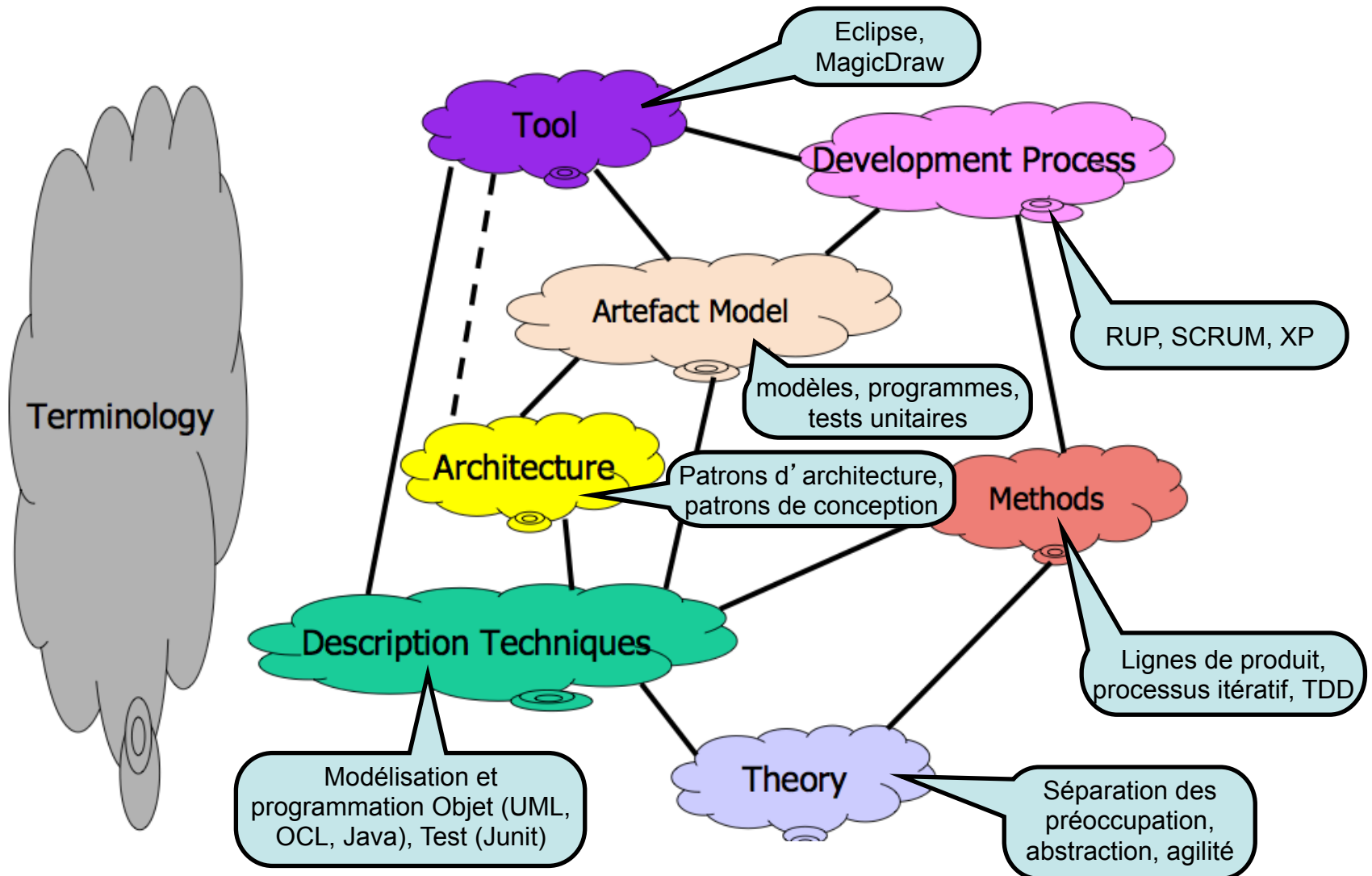
■ ...



- Phi**
- 61 networked ECUs
 - 3 bus systems + optical bus + sub busses
 - 2500 signals in 250 CAN-messages
 - more than 50 MByte memory
 - more than 2000 individual wires
 - more than 3800m cables



Software Engineering: Basics



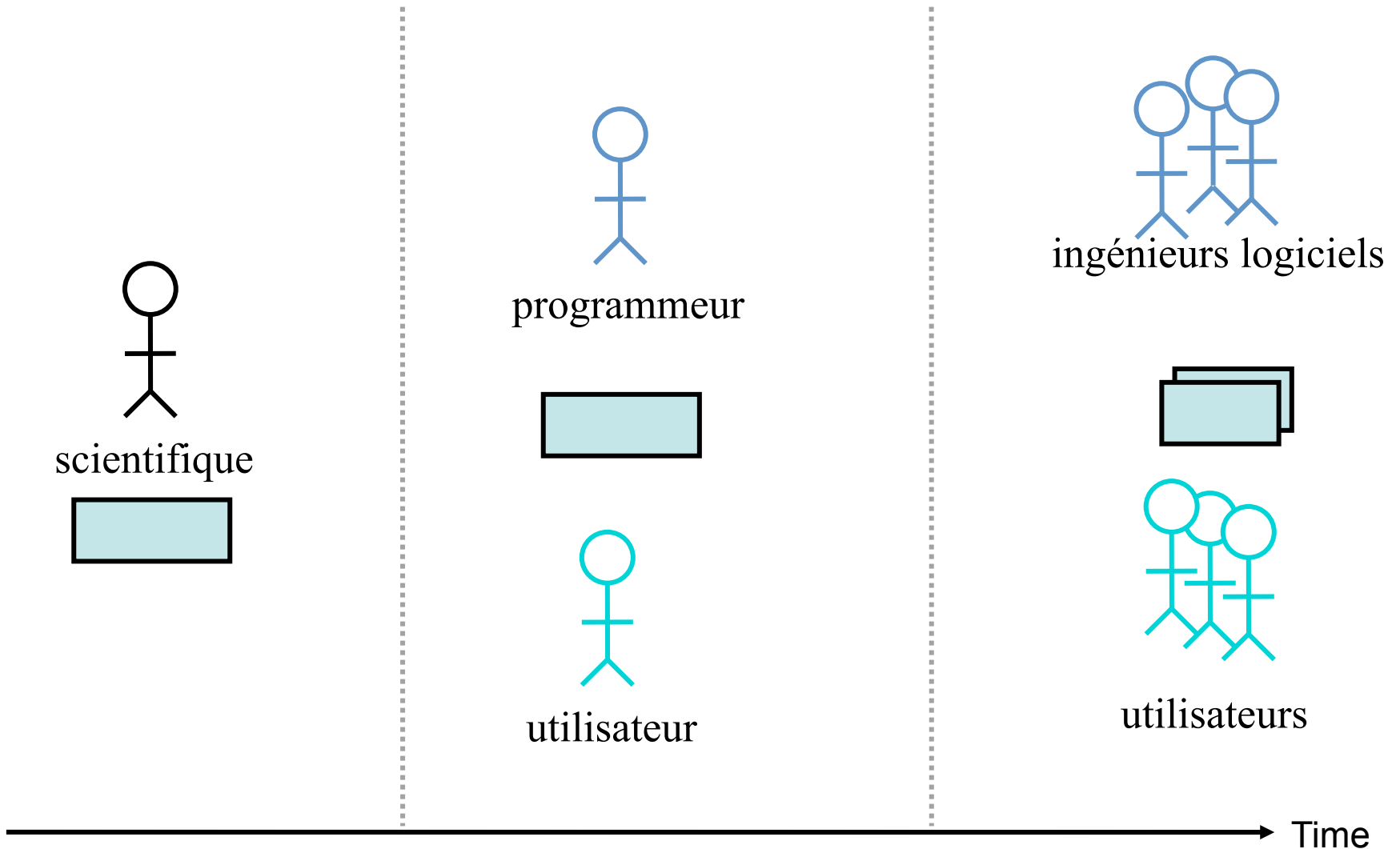
Conclusion

- **Le métier d'ingénieur logiciel est complexe:**
principes, techniques, méthodes, et outils pour décrire, implémenter, vérifier, gérer, et rendre opérationnel un système logiciel
- Réponse de l'ingénierie du logiciel par l'utilisation de la modélisation
 - séparation des préoccupations
 - montée en abstraction
 - agilité des développements

Objectifs de MDI

- Méthodes de développement industriel (MDI)
 - En fait: génie logiciel / software engineering
 - Comment développer des systèmes logiciels de plus en plus complexe?
- #1 Prendre conscience de la complexité des systèmes logiciels actuels et à venir
 - Les enjeux et l'impact sur le métier
- #2 Modélisation
 - UML, SysML
- #3 Design patterns, refactoring, test
 - OO avancé
- #4 Méthodes

Évolution des acteurs



Oui, Oui, Oui, mais ...

- La mentalité de "l'informaticien moyen" a t elle radicalement évoluée ?
- Du "Codeur", au "Programmeur", à l' "Ingénieur Logiciel", ...
- L' "Ingénieur logiciel"
 - prouve-t-il ses programmes ?
 - maintient-il à jour la documentation, les spécifications ?

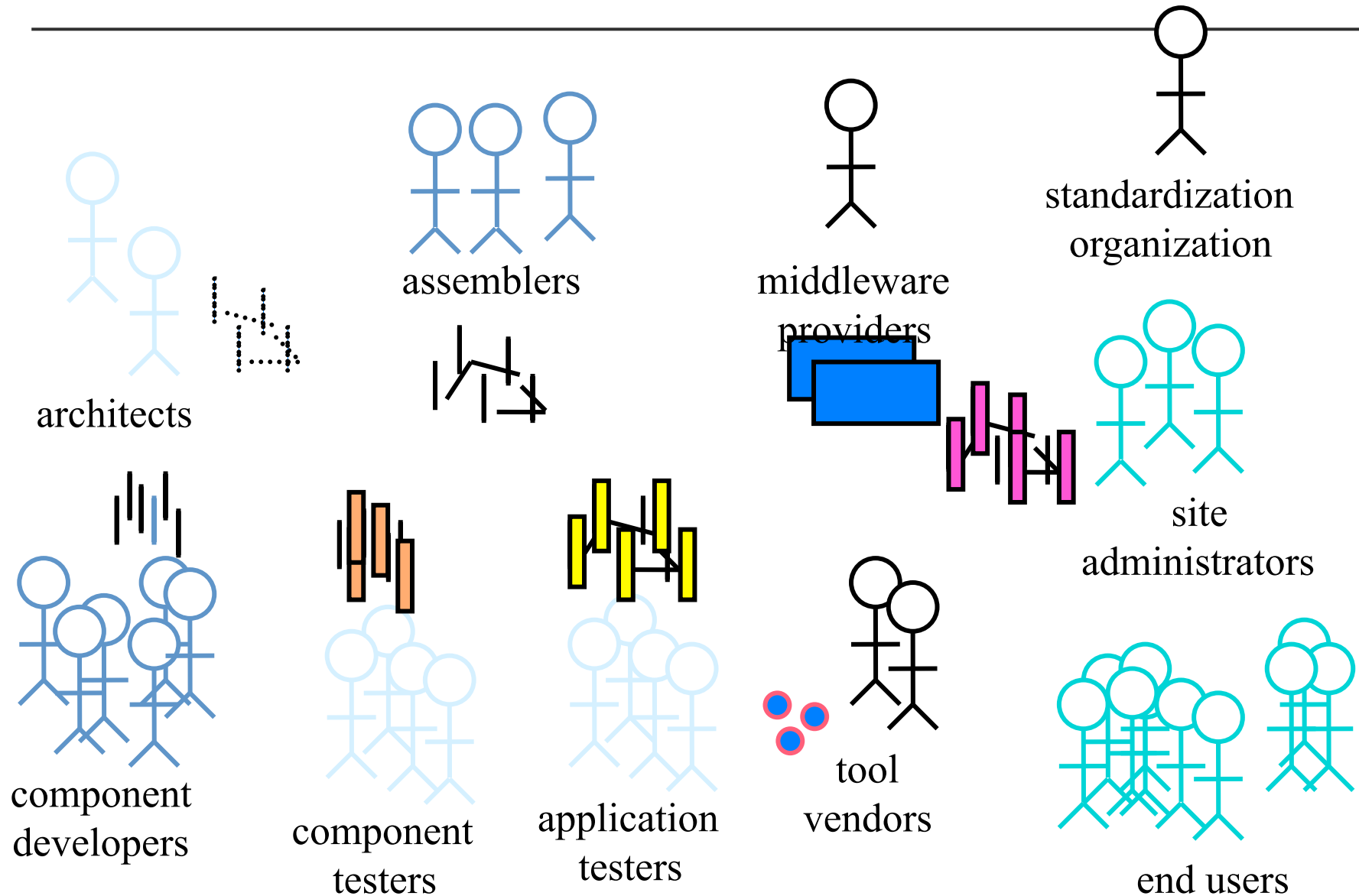
État de la pratique

50 ans après les débuts de l'informatique,
pour "l'informaticien moyen"
la seule chose importante dans le logiciel
c'est ...

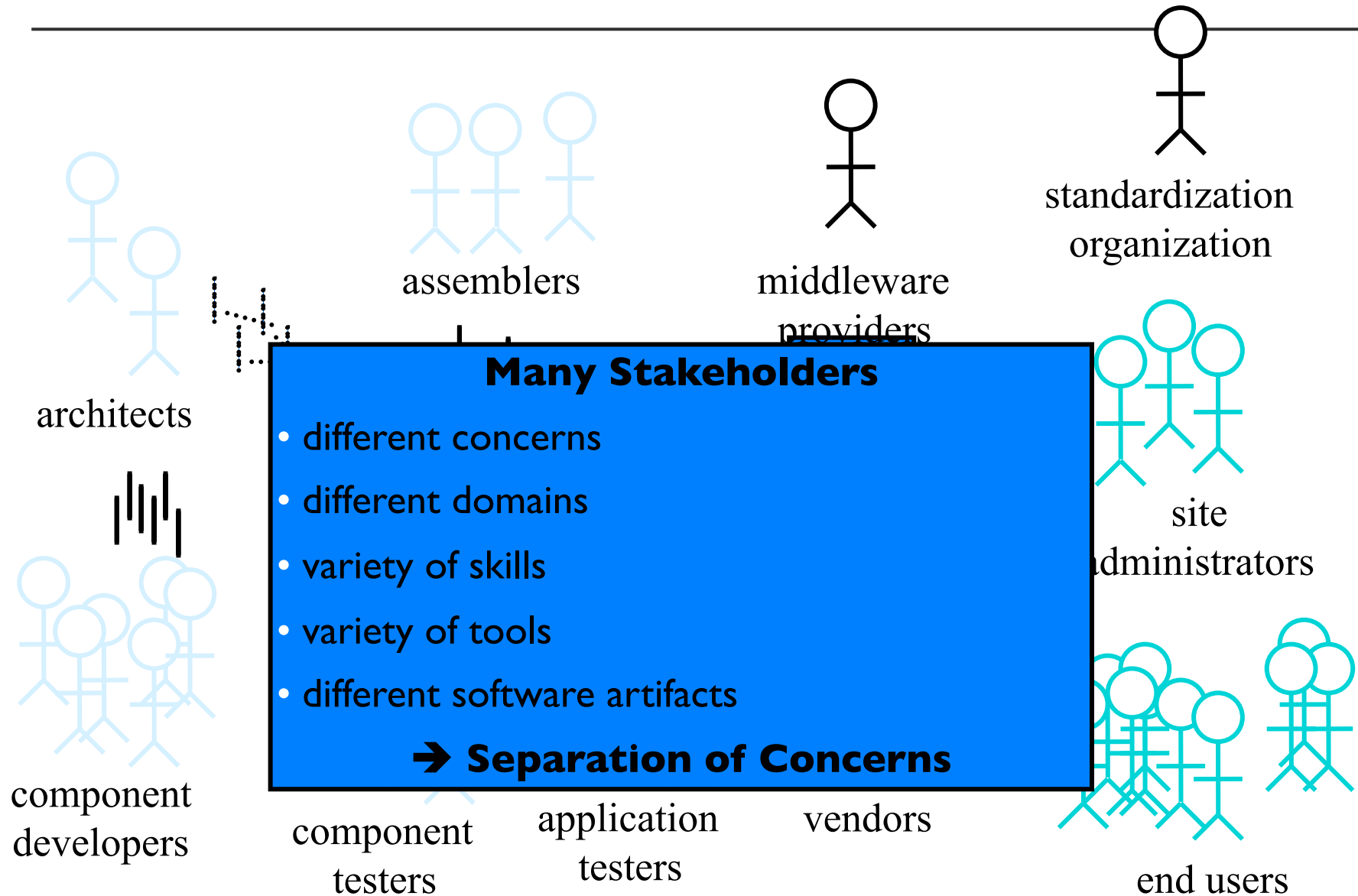
le Code !

=> Ingénierie Dirigée par le code

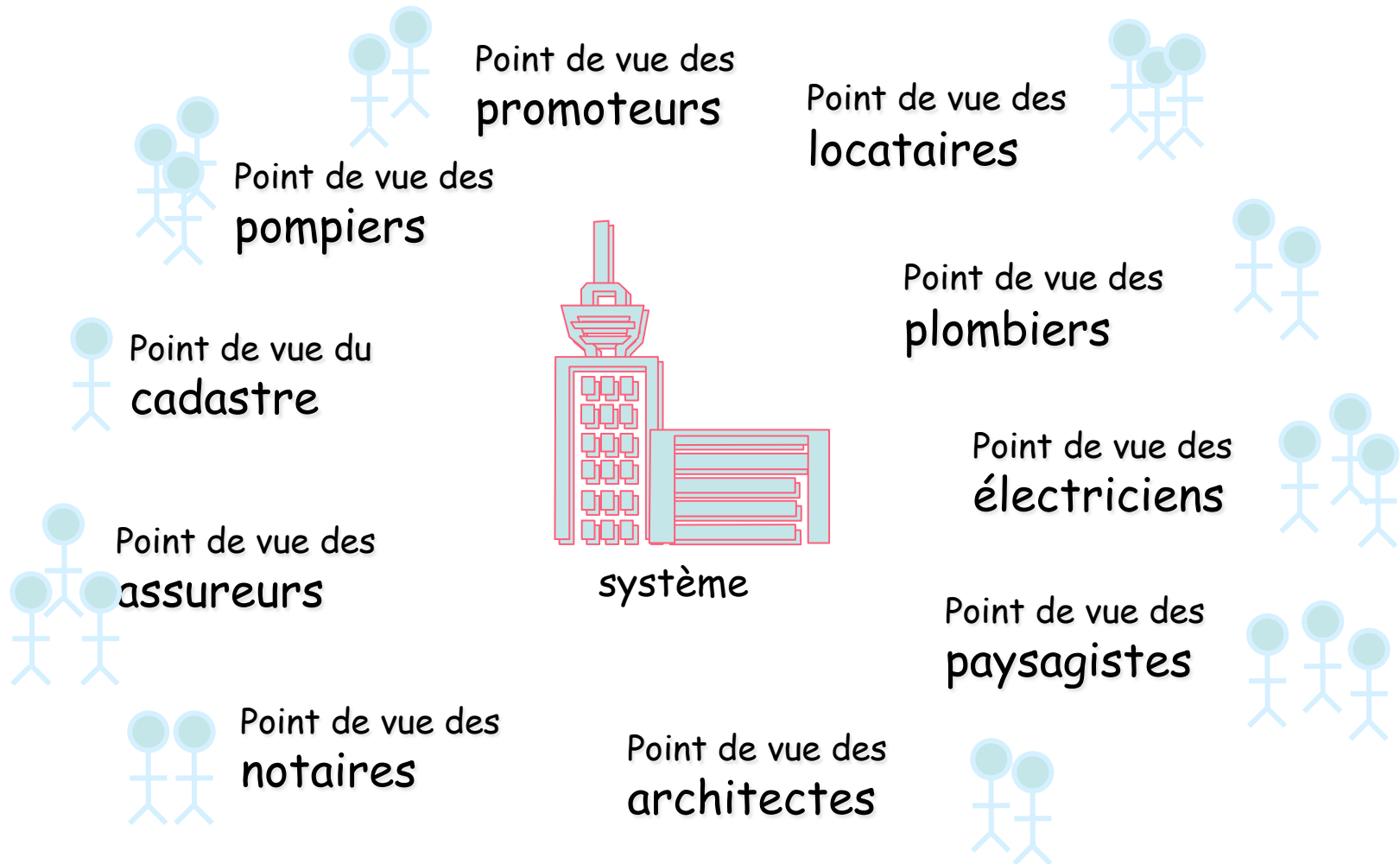
Status in Software Industry



Status in Software Industry

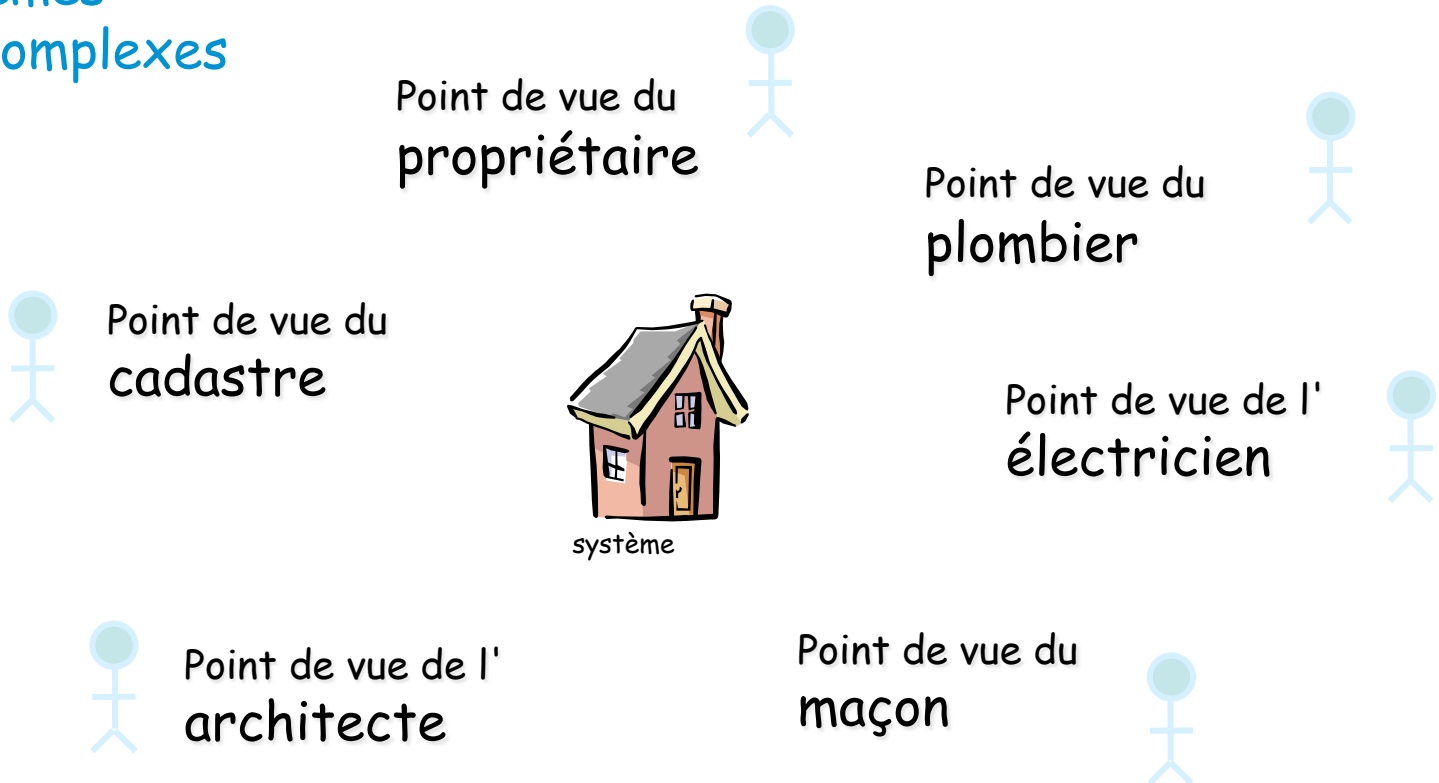


Séparations des préoccupations



Séparations des préoccupations

Utile même pour
des systèmes
"moins" complexes

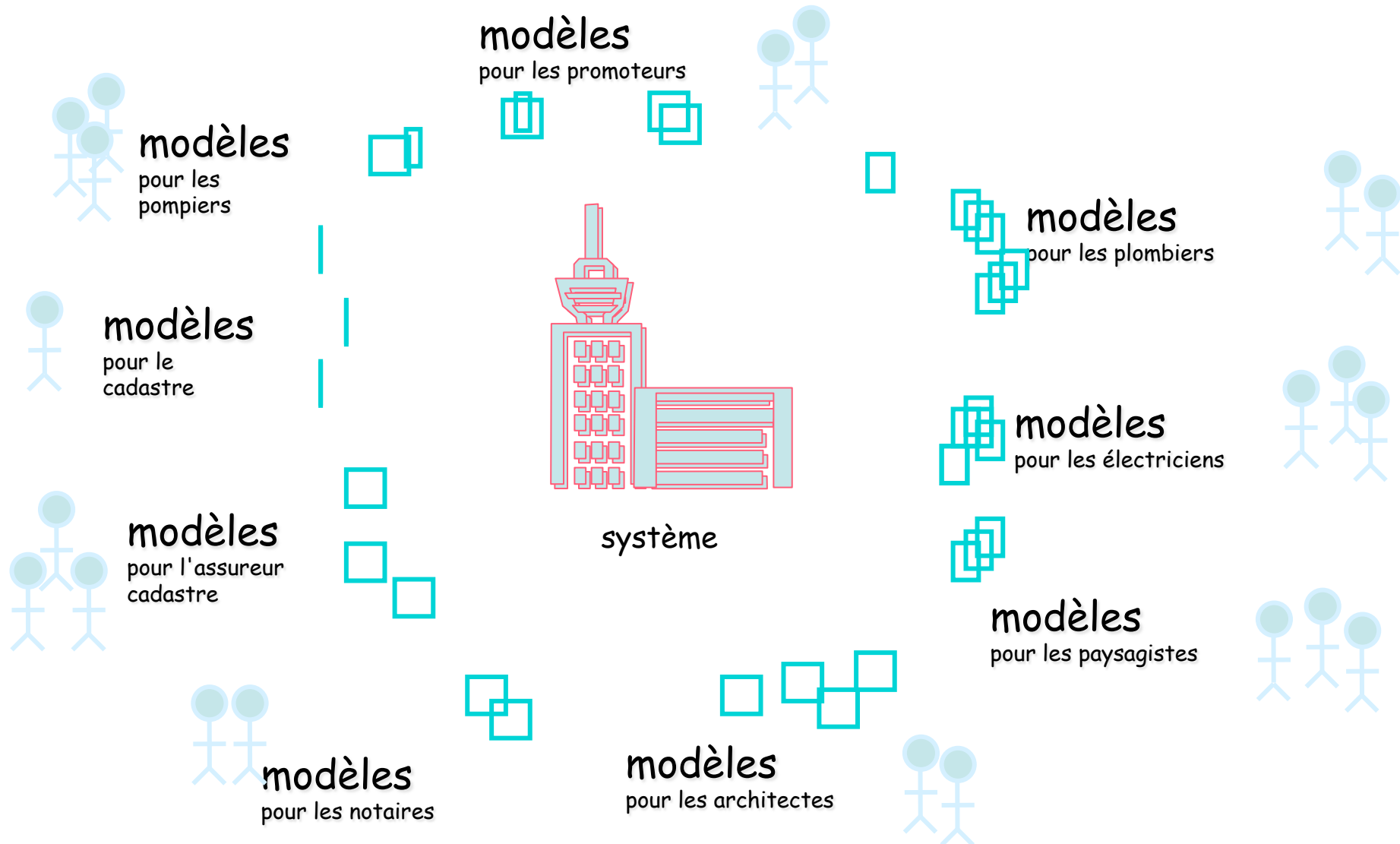


Problématique

- Complexité croissante des logiciels
- Séparations des préoccupations
- Séparations des métiers
- Multiplicité des besoins
- Multiplicité des plateformes
- Évolution permanente

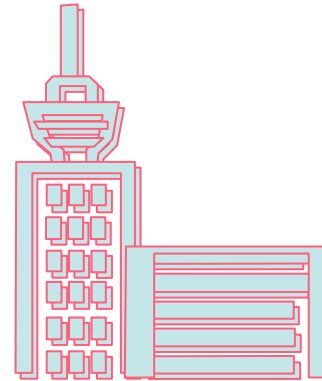
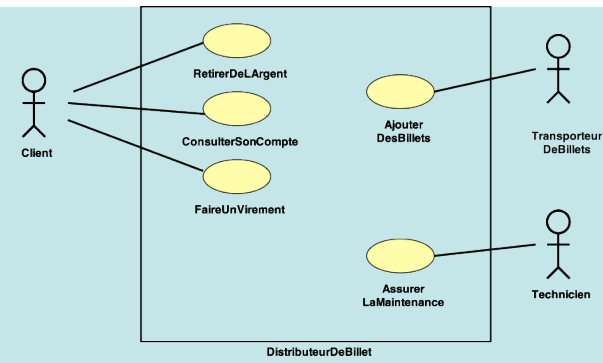
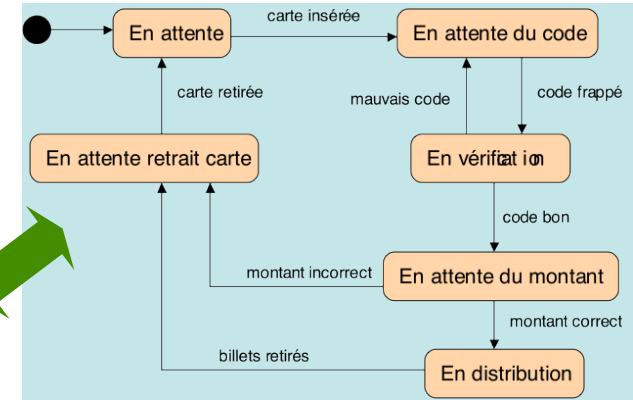
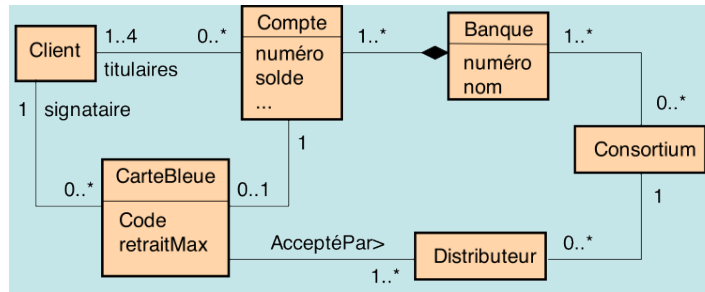
Logiciel = Code ?
Est-ce la solution ?

Multiples modèles d'un système

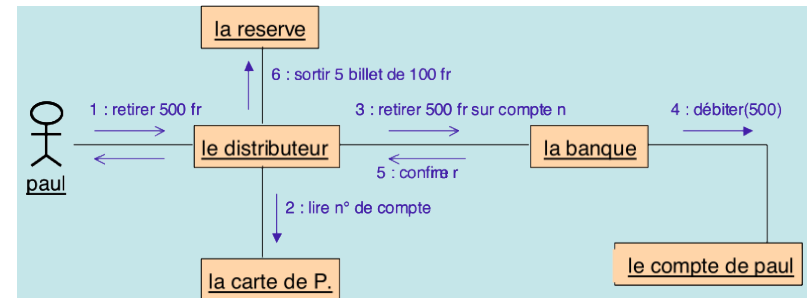
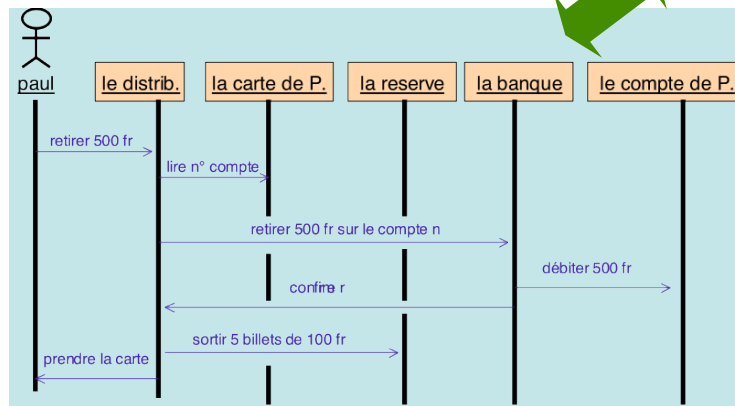


Ingénierie Dirigée par le **Code**
ou
Ingénierie Dirigée par les **Modèles ?**

Telle est la question...



système

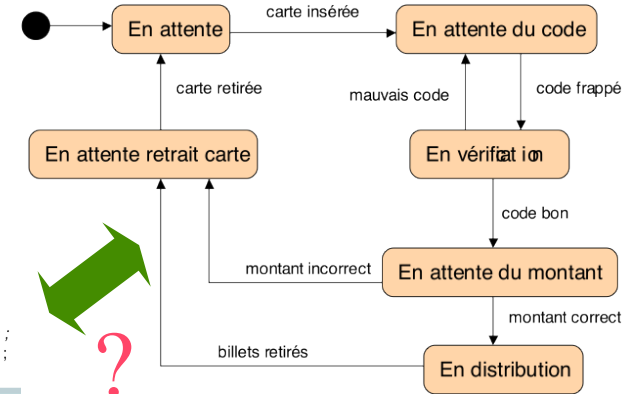
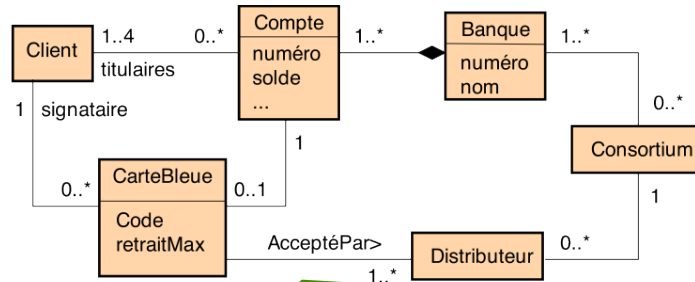


Oui, Oui, Oui, mais ...

pour "l'informaticien moyen"

la seule chose importante dans le logiciel
c'est ...

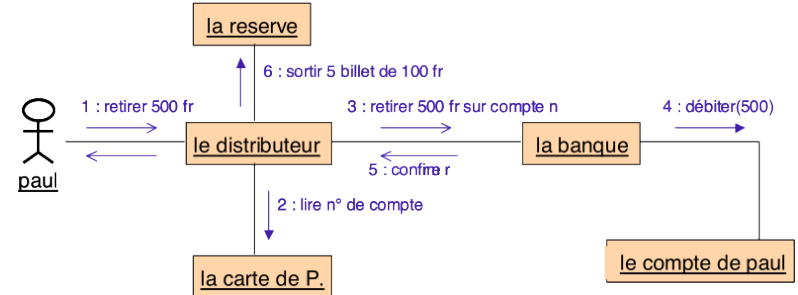
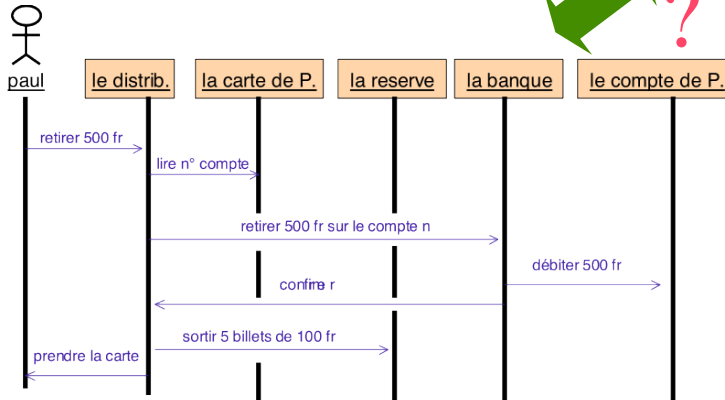
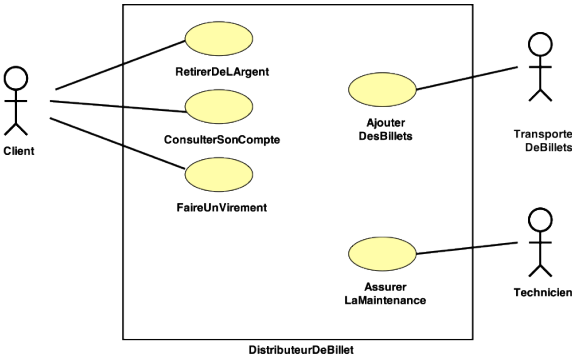
le Code !



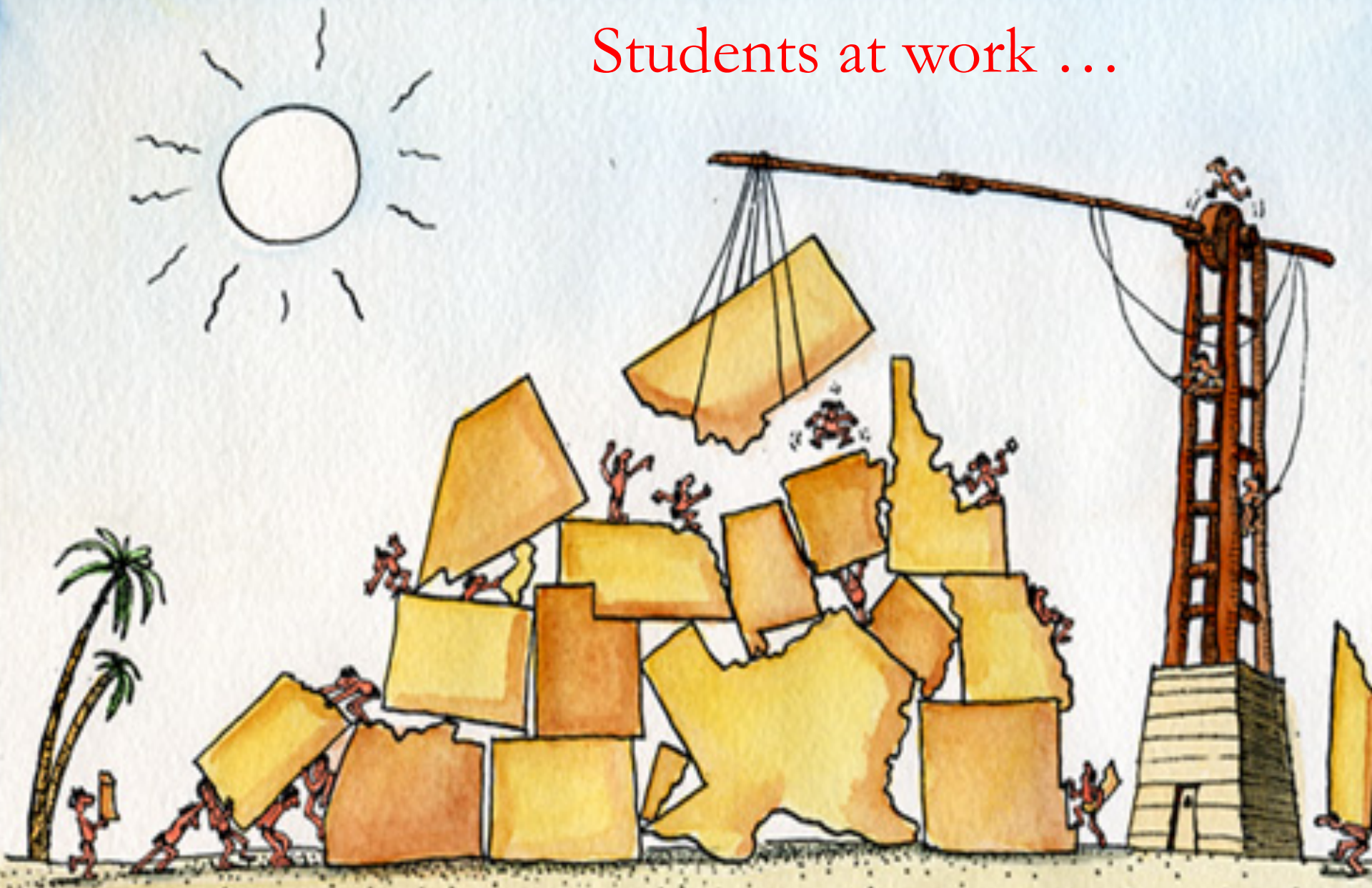
```

class Compte {
    private Vector operations ;
    private int decouvertMax ;
    private int lireSolde() { int r = 0 ; for (i=0; i<operations.size(); i++) r = r + operations.at(i).montant ; return r ; }
    public void debiter ( int montant ) { if (montant <= 0 || lireSolde() - montant < lireSolde()) throw new Exception() ; ... }
}

class Compte {
    private Vector operations ;
    private int decouvertMax ;
    private Client titulaire ;
    public int lireSolde() { int r = 0 ; for (i=0; i<operations.size(); i++) r = r + operations.at(i).montant ; return r ; }
    public void debiter ( int montant ) { if (montant <= 0 || lireSolde() - montant < lireSolde()) throw new Exception() ; ... }
}
  
```



Students at work ...



CODOR ©2008 ALL RIGHTS RESERVED

From "Teaching Programming Students how to Model: Challenges & Opportunities"
Prof. Robert B. France, EduSymp @ MoDELS, Oct. 2011

“Use of modeling techniques distinguishes a software engineer from a software developer (or programmer)”



“The earlier you start to code the longer it takes to complete the program”

**“A good modeler is a good programmer;
a good programmer is not always a good modeler”**

**“Learning a programming language is easy,
learning how to program is difficult”**

Prof. Robert B. France

Colorado State University

<http://www.cs.colostate.edu/~france/>

Des modèles plutôt que du code

- Un modèle est la simplification/abstraction de la réalité
- Nous construisons donc des modèles afin de mieux comprendre les systèmes que nous développons
- Nous modélisons des systèmes complexes parce que nous sommes incapables de les comprendre dans leur totalité
- Le code ne permet pas de simplifier/abstraire la réalité

Outline

- ① Issues in Software Engineering
- ② Evolution in Software Engineering
- ③ State of the Practice
- ④ Modeling in Software Engineering

Model to master complexity

“Modeling, in the broadest sense, is the *cost-effective use of something in place of something else for some cognitive purpose*. It allows us to use something that is *simpler, safer or cheaper* than reality instead of reality for some purpose.”

Jeff Rothenberg

The Nature of Modeling

John Wiley & Sons, Inc., August 1989

Model

“A model represents reality for a given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality.”

Jeff Rothenberg

The Nature of Modeling

John Wiley & Sons, Inc., August 1989

Exemples de modèles

- **Modèle météorologique** – à partir de données d'observation (satellite...), il permet de prévoir les conditions climatiques pour les jours à venir.
- **Modèle économique** – peut par exemple permettre de simuler l'évolution de la bourse en fonction d'hypothèses macro-économiques (évolution du chômage, taux de croissance...).
- **Modèle démographique** – définit la composition d'un panel d'une population et son comportement, dans le but de fiabiliser des études statistiques, d'augmenter l'impact de démarches commerciales, etc.

Modeling in Science & Engineering

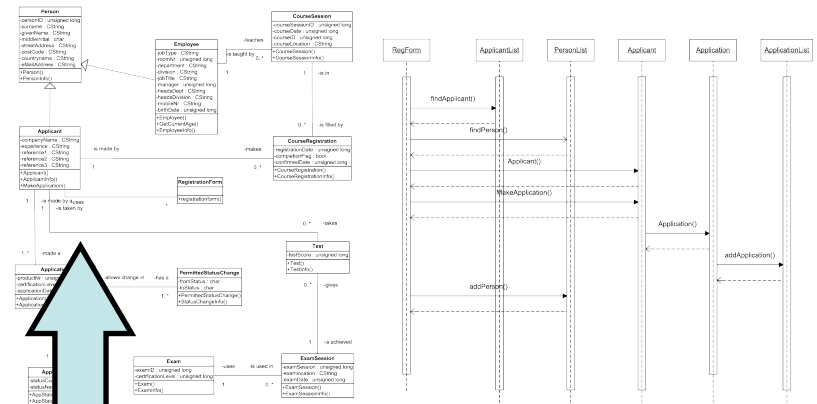
A Model is a **simplified** representation of an **aspect** of the World for a specific **purpose**

Specificity of Engineering:
Model something not yet
existing (in order to build it)

M_1
(modeling
space)

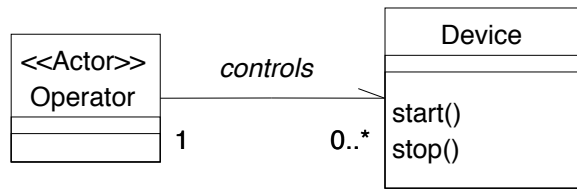
Is represented by

M_0
(the world)

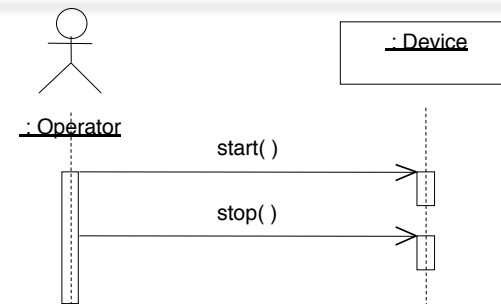


UML: one model,

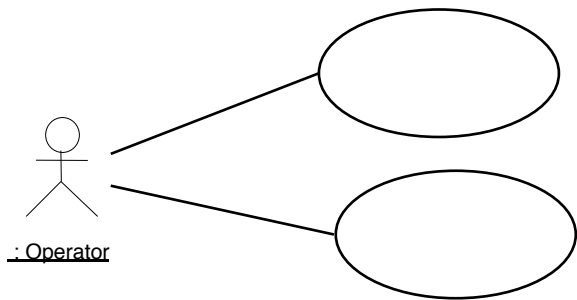
4 main dimensions, multiple views



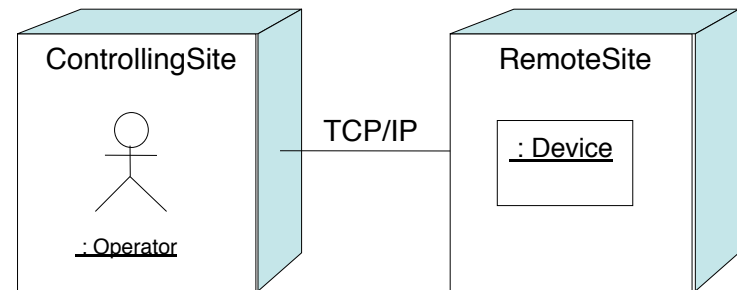
Structural view



Behavioral view

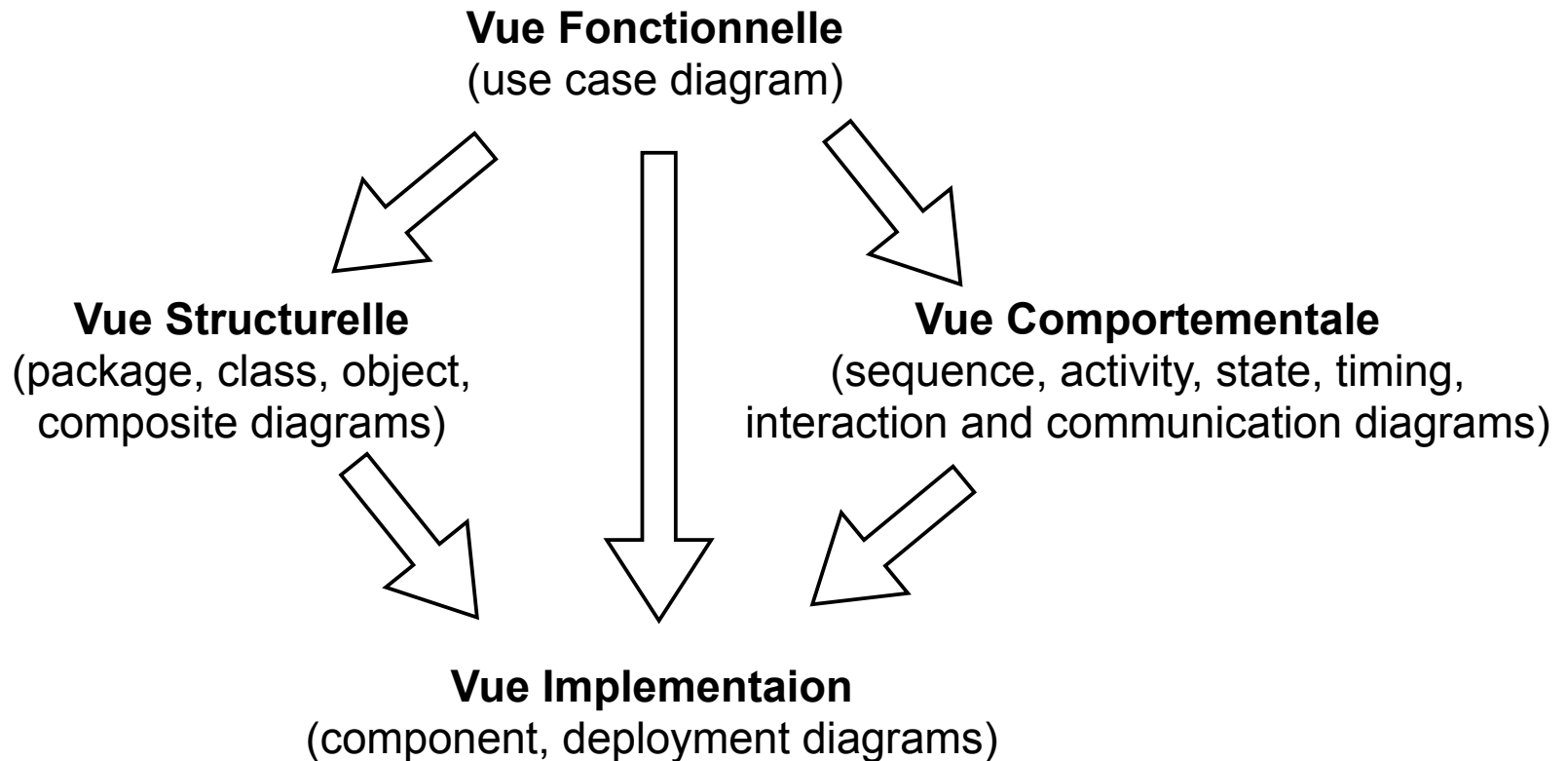


Functional view



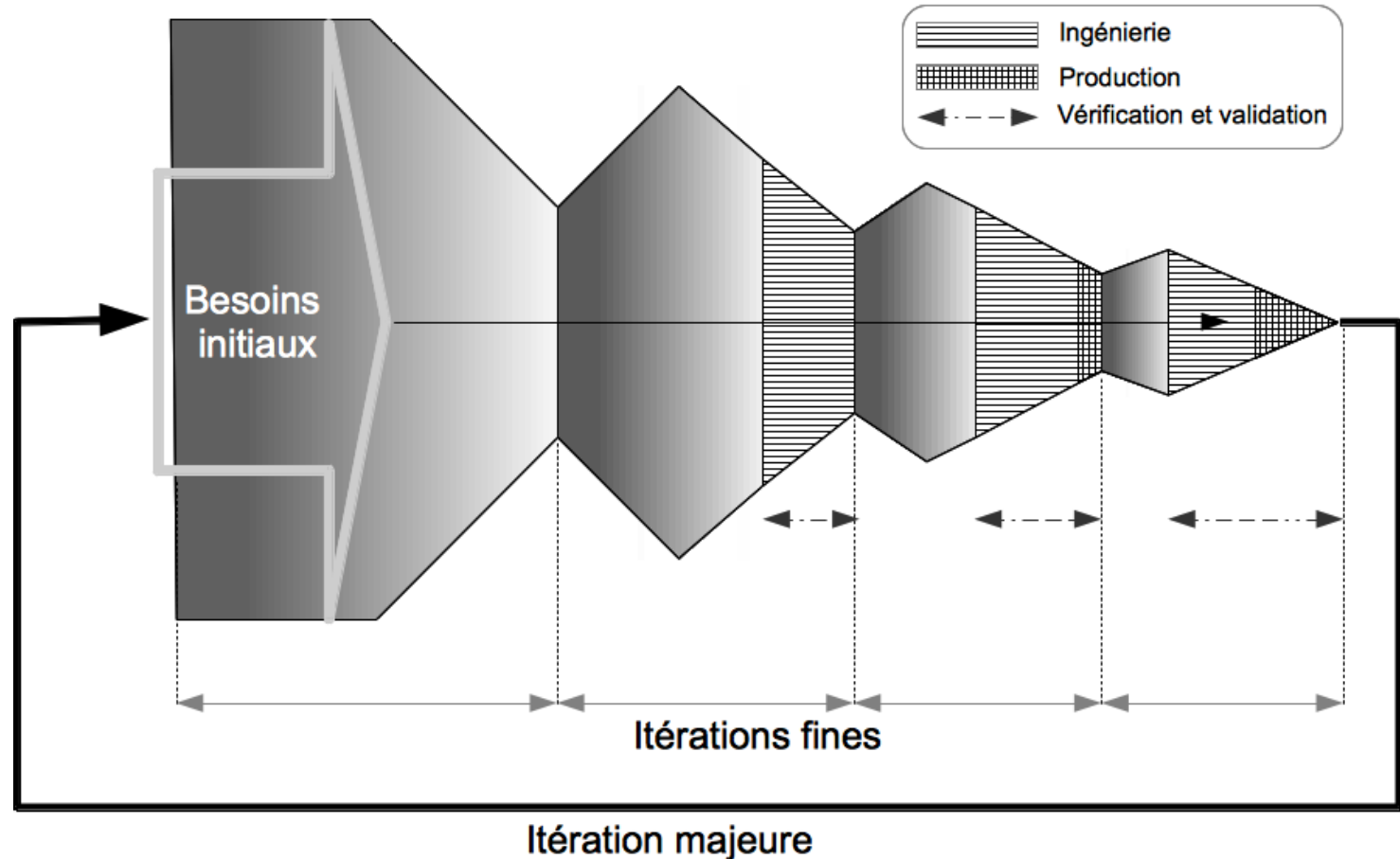
Implementation view

Une démarche sous-jacente



- La démarche doit être formalisée au sein d'un processus
- Il existe certains processus standard (e.g., RUP)

Une démarche sous-jacente



« *Sketching User Experiences: Getting the Design Right and the Right Design* » (Bill Buxton, Morgan Kaufmann, 2007)

Example

- Modeling a (simplified) GPS device

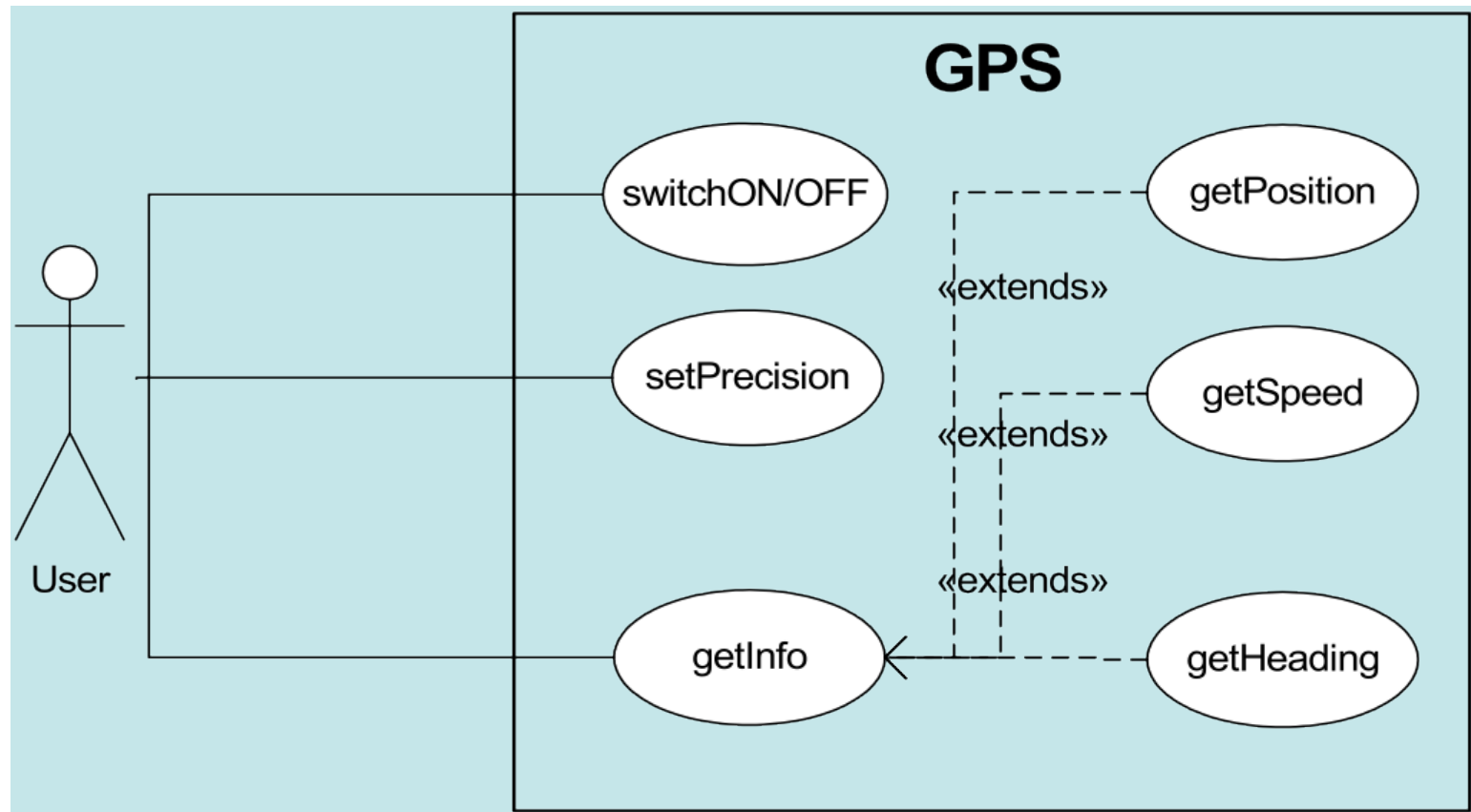
- Get position, heading and speed
 - by receiving signals from a set of satellites
- Notion of Estimated Position Error (EPE)
 - Receive from more satellites to get EPE down
- User may choose a trade-off between EPE & saving power
 - Best effort mode
 - Best route (adapt to speed/variations in heading)
 - PowerSave



*(Case Study borrowed from N. Plouzeau,
K. Macedo & JP. Thibault. Big thanks to them)*

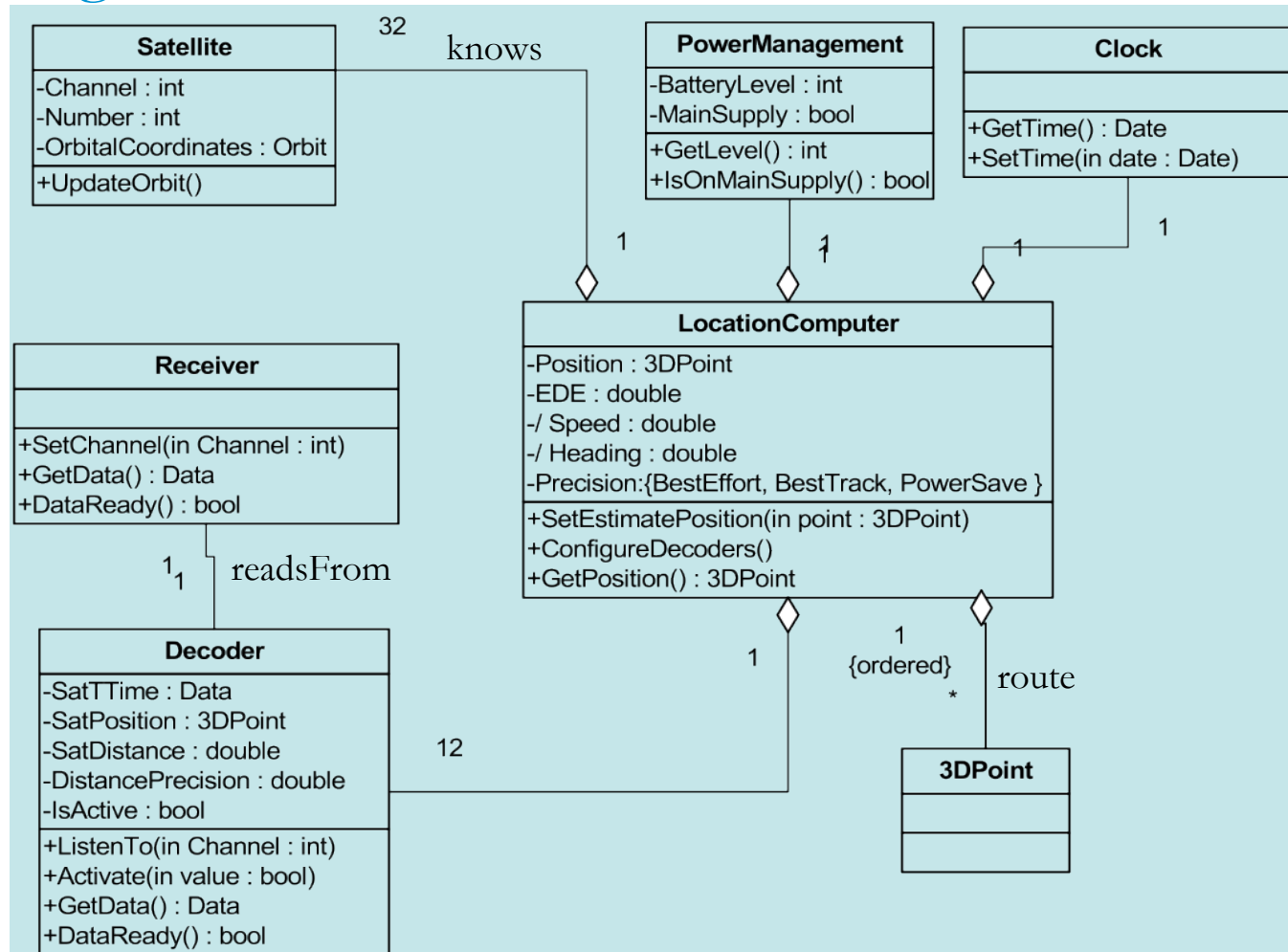
Modeling a (simplified) GPS device

- Use case diagram



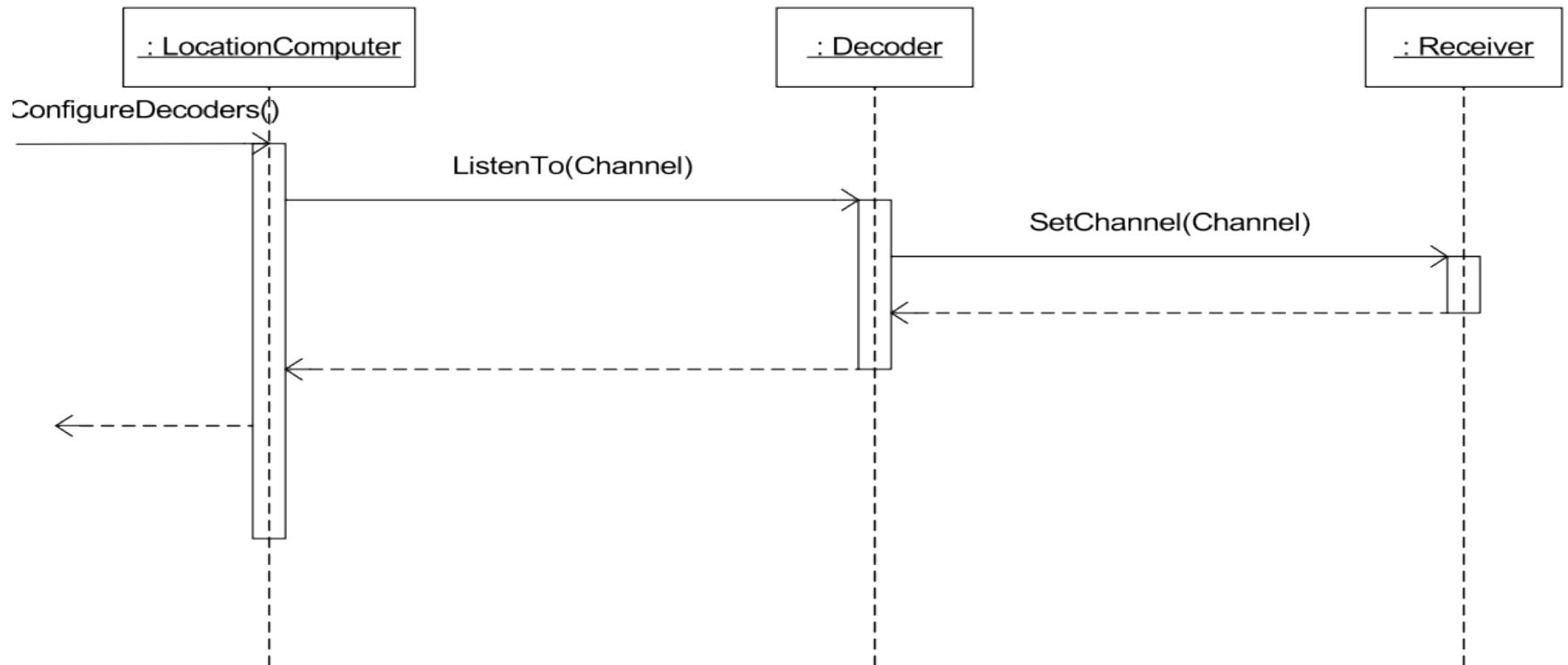
Modeling a (simplified) GPS device

- Class diagram



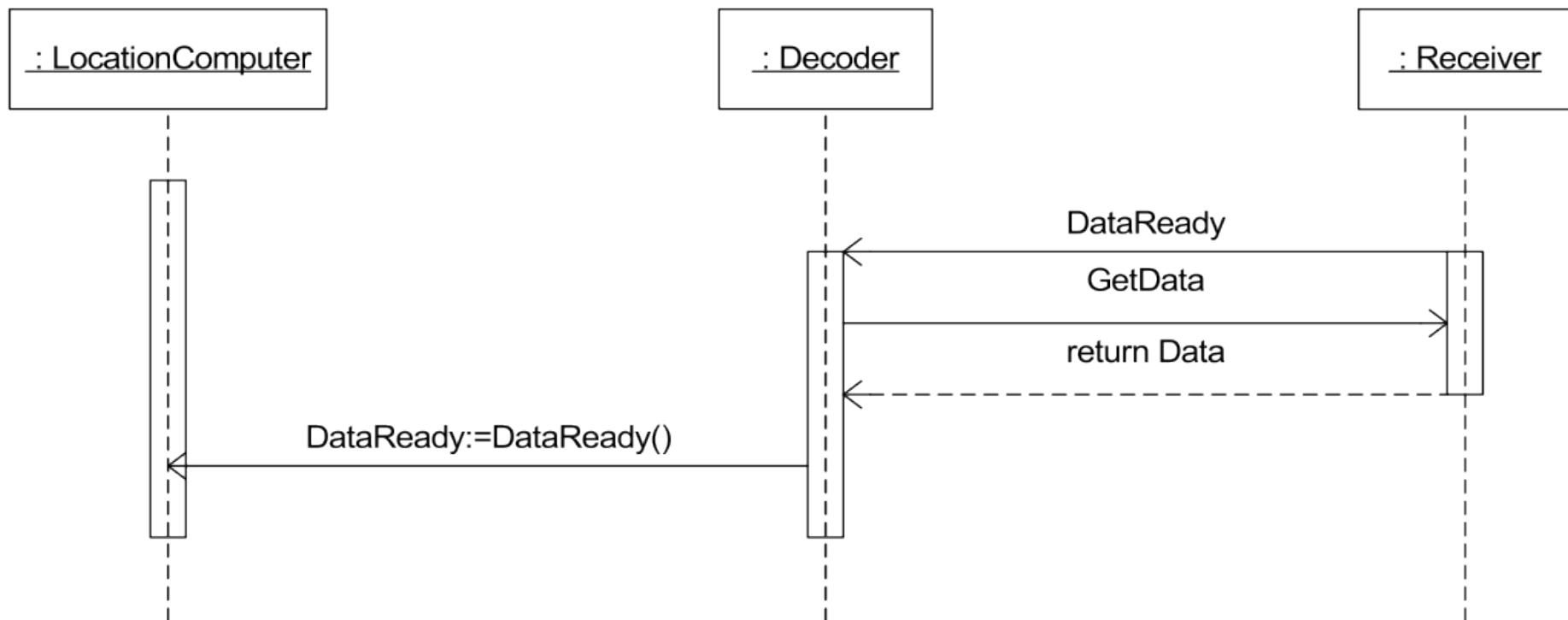
Modeling a (simplified) GPS device

- Sequence diagram: configuring decoders



Modeling a (simplified) GPS device

- Sequence diagram: interrupt driven architecture



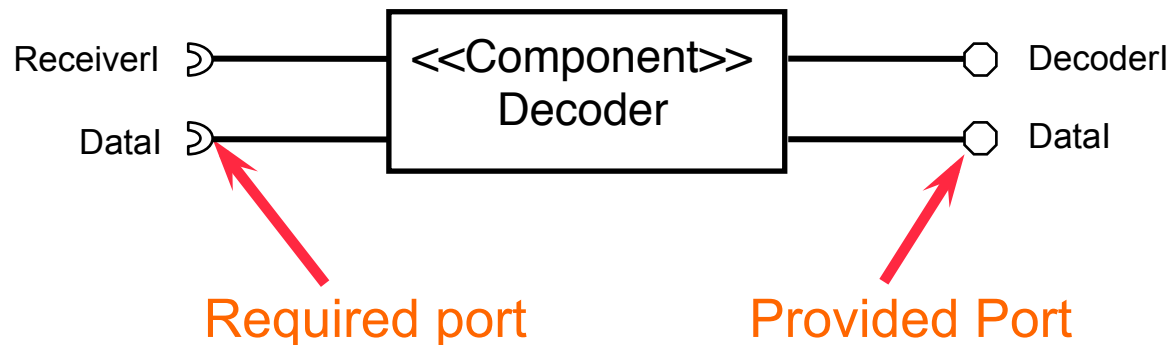
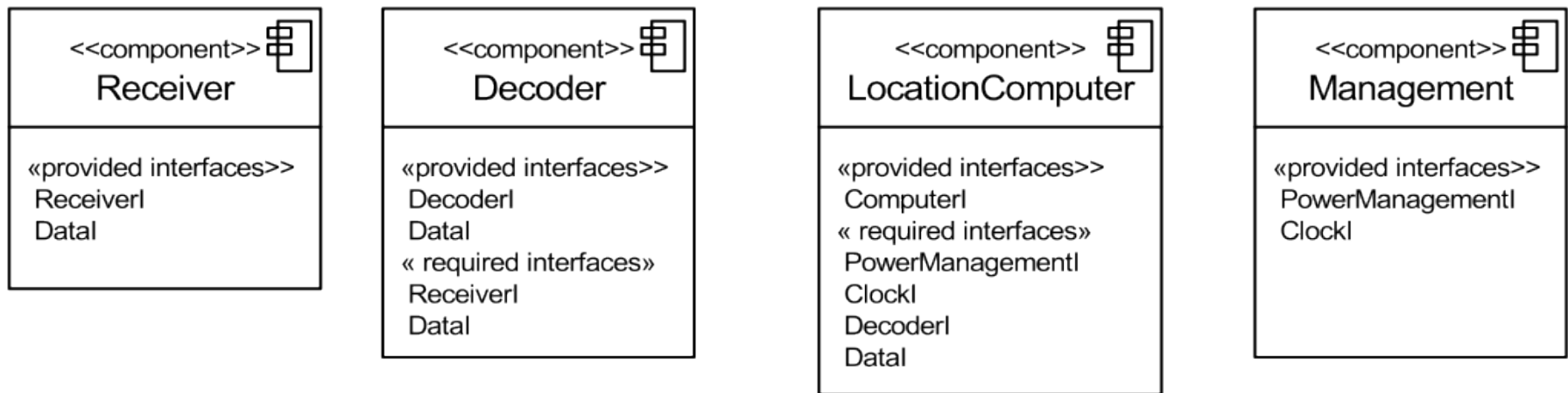
- Many more sequence diagrams needed...

Modeling a (simplified) GPS device

- Targeting multiple products with the same (business) model
 - Hand held autonomous device
 - Plug-in device for Smart Phone
 - Plug-in device for laptop (PCMCIA/USB)
 - May need to change part of the software after deployment
- ⇒ We choose a component based delivery of the software

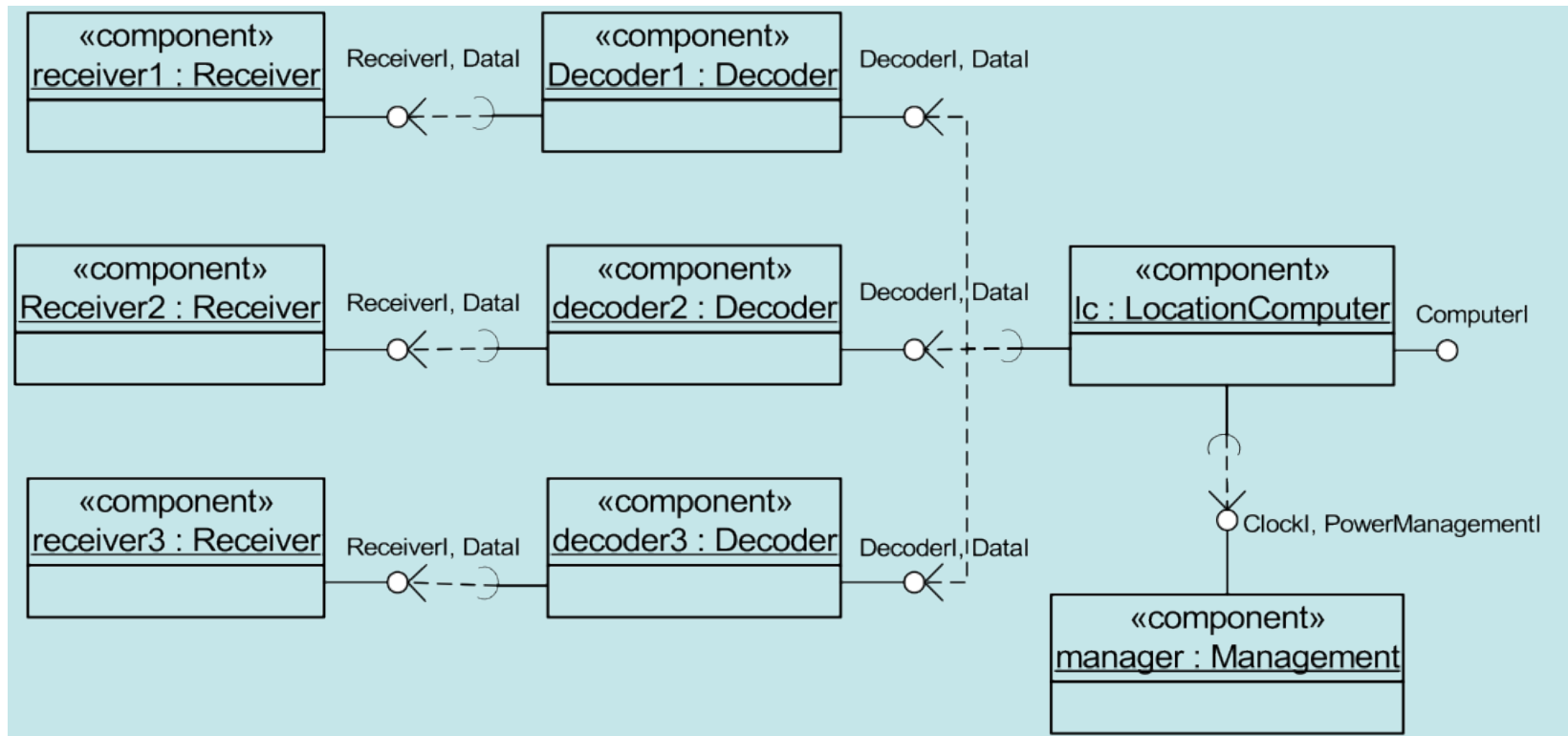
Modeling a (simplified) GPS device

- Component diagram



Modeling a (simplified) GPS device

- Deployment diagram



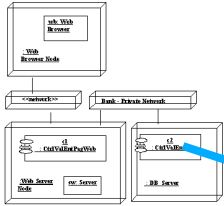
Model and Reality in Software

- **Sun Tse:** *‘Do not take the map for the reality’*
- **William James:** *‘The concept ‘dog’ does not bite’*
- **Magritte:**

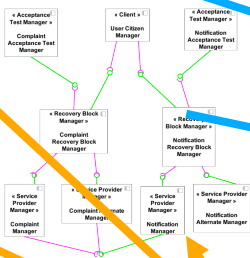


- **Software Models:** from contemplative to productive

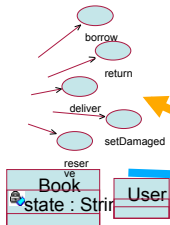
Distribution



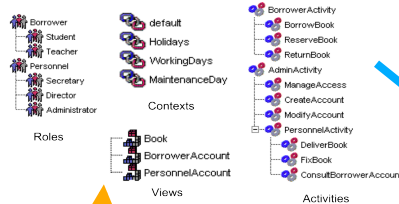
Fault tolerance



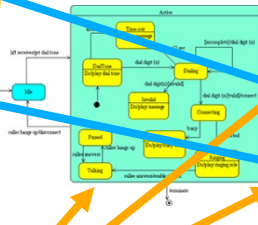
Use case model



Security



Functional behavior



Platform
Model

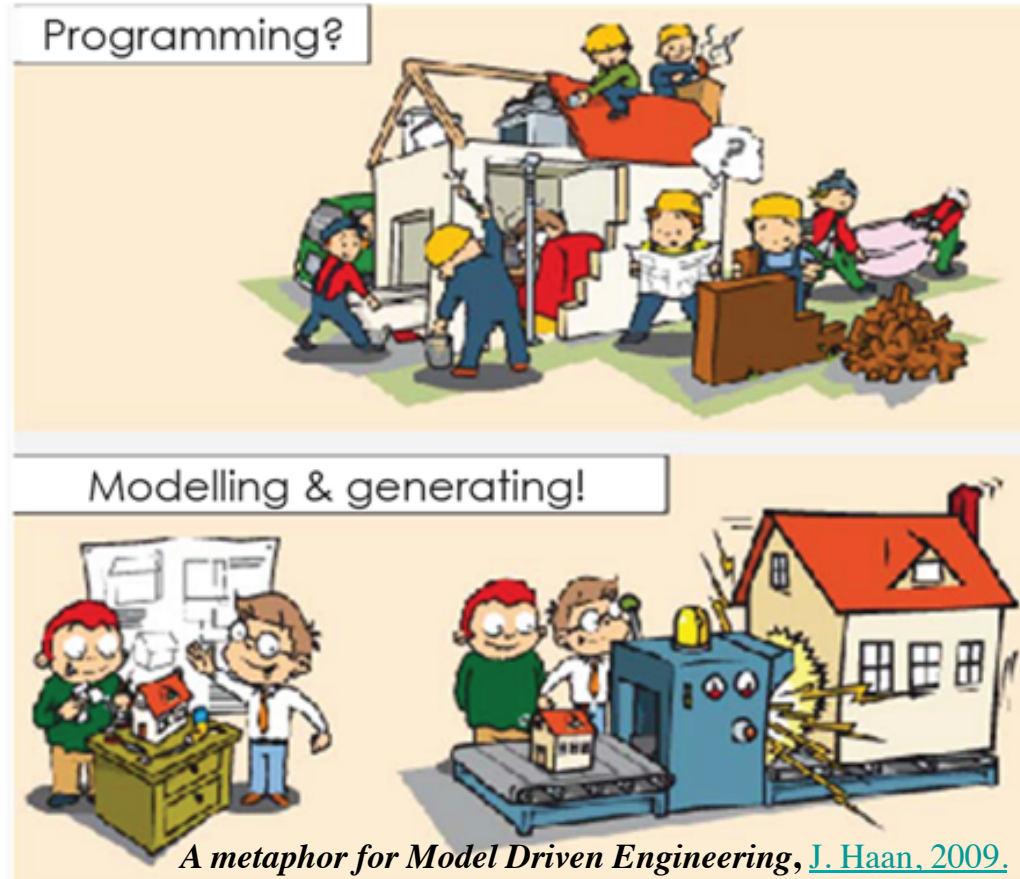
Design Model

Code Model



Change one Aspect and Automatically Re-Weave: From AORE, SPL to DAS

Towards Model Driven Engineering



La modélisation: quelle utilisation ?

- Pour réfléchir :
 - représentation abstraite
 - séparation des préoccupations
- Pour communiquer :
 - représentation graphique
 - génération de documentation
- Pour automatiser le développement :
 - génération de code
 - application de patrons
 - migration
- Pour vérifier :
 - validation et vérification de modèles (e.g., simulation, model-checking...)
 - model-based testing

Adoption of Software Modeling

